УДК 004.4.24

В.И. Межуев
Бердянский государственный педагогический университет, Украина
mejuev@ukr.net

Онтологические модели систем и процесса системной инженерии

В статье строятся онтологии систем, охватывающие различные аспекты системной инженерии, включая формулировку требований и спецификаций, архитектурное моделирование, создание рабочего плана, тестирование, валидацию и верификацию систем. Особенностью подхода является применение теории автоматов для построения онтологической модели процесса разработки систем. Рассматривается практическая реализация предложенного подхода в системе OpenCookbook.

Введение

Системную инженерию (Systems Engineering, SE) можно определить как процесс, который преобразует требования и спецификации в функционирующую систему. Изначально требования являются нечетко выраженными, так как есть результатом взаимодействия множества заинтересованных лиц – разработчиков системы. Иным важным аспектом SE является то, что разработчики выражают требования к будущей системе на естественном языке в специфичной для данной предметной области (ПрО) терминологии. Ни один из разработчиков не имеет полного представления о разрабатываемой системе вне собственной области экспертизы и часто даже не может представить, какова будет конечная система в целом. Таким образом, проблемы SE частично обусловлены фактом использования естественного языка и ограниченностью области компетентности разработчиков. Путь преодоления этих проблем состоит в моделировании и формализации процесса разработки системы. В настоящее время для моделирования систем, начальное описание которых осуществляется на естественном языке, широкое распространение получили онтологии. По определению Тома Грубера, онтологии являются точными, то есть выраженными формальными средствами, спецификациями концептуализации [1]. Следуя этому подходу, модель будущей системы мы предлагаем рассмотреть как структурированные в виде онтологий описания различных аспектов ПрО.

Онтологии предоставляют возможность выделения и формализации структуры будущей системы, описанной на естественном языке.

Постановка задачи. Для этого нами предлагается совокупность абстрактных концептов и связей между ними, соответствующих различным аспектам SE. Предлагаемая онтология SE включает такие базовые понятия, как требование, спецификация, архитектура, тестирование, верификация, рабочий план и др. Данные абстрактные концепты служат для порождения и структурирования их экземпляров – конкретных понятий разрабатываемой системы.

Цель работы. Иным важным аспектом нашей работы является определение основанной на теории автоматов модели процесса SE. В данном контексте процесс разработки системы определяется нами как машина состояний, получаемая путем наложения ограничений на переходы между этапами проектирования системы. Например, этап архитектурного моделирования может иметь место только после согласования всех

спецификаций архитектуры системы. Предложенный подход был практически реализован в web-ориентированном инструменте, названном нами OpenCookbook.

Статья имеет следующую структуру. Анализ текущих исследований и состояния проблемы представлен в следующем разделе. Раздел 2 посвящен описанию принципов предложенной онтологии и модели процесса SE. Вводятся понятия абстрактного, независимого от ПрО метаонтологического уровня и зависящего от ПрО онтологического уровня понятий. В разделе 3 описывается инструмент, практически осуществляющий предложенную методологию, а именно, определение и реализацию конкретных экземпляров процессов SE. Приводятся результаты исследований, доказывающие, что подход может быть применен к различным ПрО. Завершают нашу статью выводы и планы будущих исследований.

1 Связь с текущими исследованиями

Предлагаемый нами подход тесно связан с исследованиями, интенсивно проводимыми в других областях системной инженерии, например [2-5]. На сегодняшний день существует множество графических средств разработки систем и языков моделирования, таких как UML [6], [7] и SysML [6], [8]. Однако заметим, что эти подходы страдают от множества недостатков:

– Существующие подходы претендуют на универсальность, тогда как аспекты конкретных систем остаются вне их поля зрения. Для поддержки конкретной ПрО (или же класса ПрО) разрабатываются диалекты этих «универсальных» языков, служащие для заполнения возникающих «семантических брешей». Однако, в конце концов, эти диалекты подрывают полноценность исходной методологии. Именно поэтому мы определяем наш подход как предметно-ориентированный, что позволяет пользователю самому выбирать систему понятий и связей между ними, а также строить модель процесса определения системы.

– Большинство предлагаемых на рынке инструментов моделирования систем есть не что иное, как графический способ представления требований и спецификаций, изначально определенных в текстовой форме. Такой подход заставляет разработчиков думать в контексте предлагаемых графических нотаций и шаблонов разработки, что имеет результатом неоптимальные системные решения. Важным аспектом нашего подхода является то, что требования и спецификации не содержат в себе архитектурных решений, а также не управляются предопределенными нотациями.

– Большинство существующих подходов не имеют строгой формальной основы, а также опираются на семантически перекрывающиеся базовые понятия. Другими словами, существующим подходам недостает ортогональности в определении базовых концепций методологий моделирования ПрО. Наш подход изначально подчеркивает два аспекта SE – онтологию системы (отражающую ее структурные, функциональные и иные свойства, а также специфичные для SE требования, спецификации, рабочий план и др.) и методологию разработки системы (одним из результатов применения которой является модель процесса SE).

2 Принципы онтологии и модели процесса системной инженерии

Предлагаемая онтология SE является системой абстрактных понятий и отношений между понятиями, предназначенных для структурирования описания системы на естественном языке. Предлагаемый поход подход основан на следующих принципах:

– разграничение интенсиональных и экстенсиональных уровней в описании системы;

– различия между онтологиями и метаонтологиями (онтологиями верхнего уровня);

– построение модели процесса разработки системы;

– расширение понятия онтологии множеством методов ее использования.

Раскроем первый принцип. Предлагаемая онтология описывает систему в двух ортогональных направлениях:

1) интенсиональном (уровень намерений, в терминах SE-требований и спецификаций);

2) экстенсиональном (уровень расширений, к которому относятся архитектура системы и рабочий план по разработке системы).

Каждый из данных уровней имеет собственный набор понятий и связей между ними. Для уровня намерений, например, мы используем понятие *спецификация*, которое имеет отношение «*следует*» с соответствующим *требованием*; для архитектуры системы мы используем понятия *сущностей*, соединенных *взаимодействиями*; для планирования процесса разработки системы мы используем *задачи*, соединенные отношениями *реализации*, *тестирования*, *валидации* и др.

Раскроем второй принцип. Для каждой ПрО может быть построена собственная онтология, как система понятий и отношений между ними. Эти понятия отражают конкретные свойства, структуру, поведение данных ПрО (физических, химических, программных, аппаратных и т.д.). Нами выделяют также *метаонтологии*, которые строятся в понятиях, не зависящих от конкретных ПрО.

Такой подход сейчас широко развивается в Semantic Web [9], где онтологии подразделяются на предметно-ориентированные онтологии и абстрактные метаонтологии (или же онтологии верхнего уровня), например:

– SUMO (Standard Upper Merged Ontology) – стандартная онтология верхнего уровня [10];

– Sowa онтология верхнего уровня [11];

– Cyc'supper онтология [12] и др.

Онтологии верхнего уровня содержат так называемые утверждения здравого смысла (*common sense*) о ПрО и формируют тем самым единую для онтологий нижних уровней систему понятий.

Системная инженерия всегда имеет дело с конкретными ПрО, однако на более высоком уровне абстракции можно выделить понятия, являющиеся инвариантными для разных областей, например, объект, отношение, требование, спецификация, задача и т.д. Данные понятия связаны отношениями, также инвариантными для множества ПрО, например, часть-целое, экземпляр-класс, причина-следствие и т.д. Такие понятия высокого уровня абстракции образуют *метаонтологию* системной инженерии. Данная метаонтология используется нами для порождения онтологий конкретных ПрО.

В онтологиях понятия ПрО делятся на классы и индивидуумы (индивидуальные понятия). Предлагаемая нами метаонтология системной инженерии содержит классы, которые служат для структурирования конкретных экземпляров (индивидуальных понятий) описаний систем. Например, класс «требование» служит обобщением его конкретного экземпляра «разгон до 100 км/ч за 6 с». Таким образом, мы используем метаонтологию в качестве грамматики для описания системы на естественном языке. Понятия и отношения метаонтологии являются общими для SE и инициализируются конкретными экземплярами на уровне определения системы.

Рассмотрим третий принцип предлагаемого подхода. Отличительной особенностью онтологий является формализация связей между понятиями ПрО. Например, отношения в OWL [13] онтологиях могут иметь свойства транзитивности, функциональности, инверсности и др.

Мы также помещаем логические условия на отношения понятий метаонтологии SE, однако в нашем случае они *определяют изменение состояния системы между различными шагами в ее разработке* (рис. 1 и 2). Эти изменения состояний и есть определение *процесса* системной инженерии. Иными словами, смысл предложенного нами подхода состоит в построении модели процесса SE на основе конечного автомата, где условия отражают переходы состояний в процессе разработки системы.

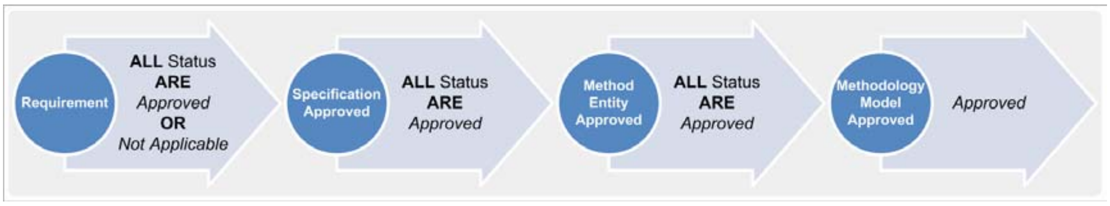


Рисунок 1 – Переходы состояний на интенсиональном уровне определения системы

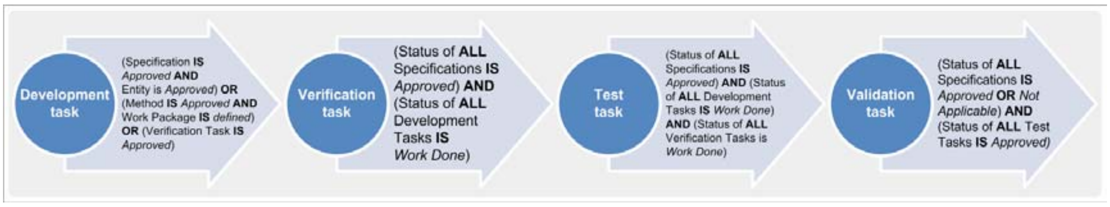


Рисунок 2 – Переходы состояний на экстенсиональном уровне определения системы

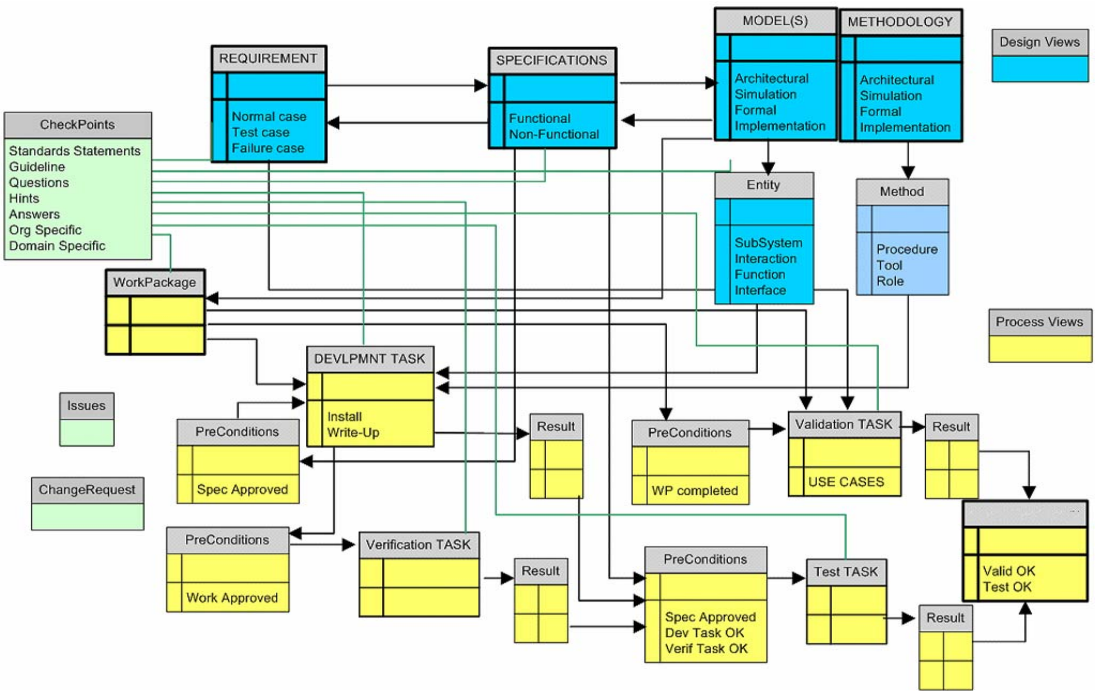


Рисунок 3 – Переходы между состояниями системы и понятиями метаонтологии

Модель процесса SE определяет, *как разработать систему правильным способом*. На рис. 3 изображены переходы между различными этапами в определении системы. Эти этапы в свою очередь являются базовыми концепциями предлагаемой метаонтологии SE.

Раскроем последний принцип предлагаемого нами подхода – расширение модели онтологии множеством методов ее использования.

Онтология O ПрО d (от англ. *domain*) определяется как набор понятий и отношений между понятиями $O^d \equiv \langle X, \Re \rangle$.

$X = \{x_l \mid l = 1,...,L\}$ – конечное множество понятий ПрО.

$\Re = \{r_k \mid k = 1,...,K\}$ – конечное множество отношений между понятиями.

Онтология может также включать иные элементы, например аксиомы, ограничения, правила вывода, функции интерпретации и т.д.

Идея состоит в возможности императивного использования декларативного подхода, который предоставляют онтологии. Данный подход мы осуществляем путем добавления в определение онтологии множества методов ее использования M .

$M = \{m_n \mid n = 1,...,N\}$ – конечное множество методов использования онтологии.

Таким образом, мы определяем онтологию как: $O^d = \langle X, \Re, M \rangle$.

Приведем примеры методов, применимых к онтологии SE:

– метод проектирования системы (определяющий порядок этапов определения системы в процессе SE);

– валидация и верификация свойств системы (например, проверка корректности вложенности архитектурных элементов системы);

– стандартизация системы (проверка соответствия некоторым общепринятым правилам);

– генерация архитектуры системы (например, диаграммы классов, в случае построения онтологии программной системы);

– генерация кода программного кода (в случае, когда онтология описывает структуру программного приложения) и др.

2.1 Онтология интенсионального уровня

Рассмотрим подробнее основные понятия и отношения интенсионального уровня определения системы. Как было замечено, системная инженерия – это процесс, который преобразовывает исходные требования в рабочую систему. Первоначально мы определяем систему с точки зрения наших намерений. Эта точка зрения и есть интенсиональный уровень определения системы. Следующий уровень – экстенсиональный – определяет, как система должна быть реализована (т.е. архитектуру и методы разработки).

С точки зрения интенсионального уровня *система* проектируется для того, чтобы выполнить все выдвинутые требования. Для этого система декомпозируется на элементы (в системной инженерии называемыми также модулями или подсистемами). В предлагаемой онтологии мы называем элементы *сущностями*, а способ, которым они связаны друг с другом, мы вызываем *взаимодействием*. Заметим, что каждая сущность может быть также отдельной системой; иными словами, понятие сущности является иерархическим.

Термин «система» используется нами в том случае, когда взаимодействующие сущности демонстрируют свойства, которыми они не обладают в отдельности. Например, самолет можно определить как систему взаимодействующих сущностей (корпус, крылья, шасси и т.д.), которые в отдельности стремятся к падению, однако могут лететь как целое. Как сущности и взаимодействия образуют архитектуру системы, так отдельные требования формируют предназначение системы в совокупности.

Мы делаем явное различие между требованиями и спецификациями. Спецификации являются измеримыми экземплярами требований и связываются нами с возможностью их проверки (путем тестирования, валидации, верификации и других методов). Можно иметь несколько систем с общими требованиями, но различными спецификациями (например, в зависимости от граничных условий, таких как стоимость). Следо-

вательно, предпосылкой для проектирования системы (перехода с интенсионального на экстенсиональный уровень) являются спецификации, но не требования непосредственно.

Заметим, что понятия требования и спецификации часто не различаются, и это приводит к недоразумениям. Также используется весьма неоднозначное сочетание «спецификация требований». В нашем подходе мы используем термин «требование» для обозначения желаемых свойств системы. Спецификации являются количественно определенными требованиями с ассоциированными методами тестирования и проверки.

Формулировка требований и спецификаций является самой важной частью процесса описания системы. Спецификации порождаются путем конкретизации более общих требований и указания процедуры тестирования. Например, изначальное требование «автомобиль должен быть быстрым», может быть преобразовано в спецификации «ускорение от 0 до 100 км/ч за 6 с» и «достижение максимальной скорости не менее 200 км/ч». Процедура тестирования в данном случае сводится к измерению величин скорости и ускорения.

Заметим, что спецификации часто формулируются со скрытым предположением, что система работает без наблюдаемых или латентных проблем. Мы называем такие спецификации *нормальными случаями (normal case)*. Однако этого недостаточно. Система соответствует спецификациям только после прохождения *тестовых случаев (test case)*, описывающих процедуры проверки спецификаций. В соответствии с тестовыми случаями мы определяем *случаи отказа (fault cases),* то есть последовательности действий, которые могут иметь результатом отказ или некорректное поведение системы. Идея состоит в том, что, формулируя случаи отказа, мы начинаем осознавать, что может пойти неправильно. Понимание этого *прежде* реализации системы позволяет предупредить многие возможные бедствия и катастрофы.

Таким образом, использование онтологии для описания системы на интенсиональном уровне позволяет нам создать концептуальную модель из первоначально разрозненных и нечетких требований разработчиков. Требования, спецификации, нормальные случаи, тесты, случаи отказа есть не просто набор понятий, а концептуальная модель системы со структурой, определенной отношениями онтологии.

2.2 Онтология экстенсионального уровня

В экстенсиональном или же архитектурном представлении система определяется *сущностями* и *взаимодействиями* между ними. Сущность имеет собственные *атрибуты и функции*. Атрибут является внутренней характеристикой сущности и отражает ее качественные и количественные характеристики (например, цвет, скорость, размер и т.д.). Атрибуты характеризуются именем, типом и значением. Имя является необходимым атрибутом всех сущностей.

Функции определяют внутреннее поведение сущности в отличие от их внешних взаимодействий. Можно указать различные подходы к определению онтологии экстенсионального уровня. Это может быть функциональный, структурный, причинный, событийный подходы, рассмотрение системы как машины состояний и др.

Рассмотрим, например, совокупность понятий онтологии, отражающей событийный подход. В этом контексте взаимодействия вызываются *событиями* и реализуются как последовательность *сообщений* между сущностями системы. Сообщение может быть вызвано событием и может также вызывать события. Такой подход имеет широкое распространение в моделировании параллельных программных систем, где взаимодействие подразумевает некоторую форму передачи сообщений между объектами. Такие сообщения могут использоваться для пересылки данных или же вызывать

соответствующие функции внутри программной сущности. Вообще говоря, событием может быть любое изменение, имеющее место в системе. Например, событие может быть результатом изменения атрибута сущности, влекущего изменение состояния всей системы. Мы также предполагаем, что взаимодействие изменяет состояние всех участвующих в нем сущностей.

Иным важным понятием предлагаемой онтологии экстенсионального уровня является *интерфейс*. Интерфейс является точкой взаимодействия между двумя или более сущностями. Интерфейсы могут иметь *тип вход (input)* или *выход (output)*, определяющий направление передачи материи, энергии или информации в процессе взаимодействия сущностей. Примеры интерфейсов – электрическая розетка, топливный конвейер, порт USB и т.д.

Интерфейсы необходимы для осуществления взаимодействий между сущностями. Интерфейс связывает внешнее взаимодействие с внутренними для сущности функциями. Будучи инициатором взаимодействия, интерфейс сущности преобразует внутреннее событие во внешнее сообщение. Субъект взаимодействия также получает сообщение через собственный интерфейс, преобразовывая внешнее сообщение во внутреннее представление (событие). Таким образом, интерфейс фильтрует полученные сообщения и вызывает адекватные сущности функции. Пересылка данных в программной системе является самым простым примером подобного рода взаимодействий. Заметим также, что носитель взаимодействия также является системой со своей собственной структурой. Физические, логические и др. свойства носителя также влияют на поведение системы. Как пример приведем магистрали Интернета, гидравлические каналы, линии электропередач и т.д.

2.3 Связь между интенсиональным и экстенсиональным уровнями в определении системы

Как было замечено выше, на интенсиональном уровне система определяется требованиями. В дальнейшем требования должны быть преобразованы в экстенсиональные архитектурные определения, осуществляемые в понятиях сущность, взаимодействие, атрибуты, функции и др. В свою очередь, данные определения ведут к изменению интенсионального уровня, т.е. уточнению начальных требований и спецификаций.

Выделение архитектурных атрибутов сущности начинается еще на интенсиональном уровне. Например, требование «достижение автомобилем максимальной скорости не менее 200 км/ч» декомпозируется в архитектурную сущность («автомобиль»), который имеет атрибут типа скорость (определяемый единицей измерения, в данном случае «км/ч») с целочисленным значением «200».

Переход от интенсиональных требований к экстенсиональному архитектурному уровню достигается декомпозицией, типификацией, структурированием, определением иерархии и другими методами. Качественные интенсиональные требования производят экстенсиональные сущности, взаимодействия, интерфейсы, атрибуты, функции, то есть архитектурные описания элементов. В свою очередь, архитектура влияет на спецификации системы (нормальные случаи, тестовые случаи, случаи отказа), план работы, а также список нерешенных проблем. Этот порядок действий является существенным и задает *модель процесса* определения системы (рис. 3).

Заметим, что на интенсиональном этапе процесса проектирования систем еще не существует точного архитектурного разложения на сущности и их взаимодействия. Существует только незавершенная концептуальная модель, которая выражается в форме требований и спецификаций. Задача системного инженера состоит в том, чтобы пре-

образовать ее в экстенсиональную форму, то есть разработать архитектурную модель, которая будет изоморфной реальной системе.

В качестве метода перехода с интенсионального на экстенсиональный уровень мы предлагаем следующий подход. Во время интенсионального этапа мы выделяем номинальные *идентификаторы* сущностей, которые будут использоваться на экстенсиональном уровне при архитектурном моделировании и планировании работ. Причина этого состоит в том, что определение архитектуры на интенсиональном этапе номинально, то есть мы можем формировать и оперировать только тезаурусом имен сущностей и взаимодействий, но не реальной архитектурной моделью. Поэтому данный метод можно рассмотреть как первый шаг к переходу от интенсионального к экстенсиональному уровню определения системы.

Интересной проблемой является исследование сущности интенсиональных и экстенсиональных отношений между понятиями соответствующих онтологий. Эти отношения отличны по своей природе (например, отношение субординации между требованиями не подразумевает, что такое подчинение существует между соответствующими архитектурными сущностями). Вообще говоря, разработка метода перехода от интенсиональной модели требований к экстенсиональной архитектурной модели является весьма сложной задачей. Эта проблема сводится к поиску архитектурных отношений в интенсиональной модели, отражающих структурные, функциональные, временные и иные соотношения реальной системы.

2.4 Планирование, верификация, тестирование и валидация

Другой важный аспект процесса системной инженерии – планирование разработки проекта, процесс которого в нашем подходе также основан на *идентификаторах*, являющихся результатом архитектурной декомпозиции системы. После идентификации сущности группируются в *пакеты работ (work packages)*, которые используются нами для планирования проекта. Каждый пакет работ делится на *задачи* с атрибутами, такими, как продолжительность, ресурсы, конечные сроки, ответственные лица и т.д. Также рассматриваются запросы на изменение проекта (*change requests*). Некоторые задачи связываются не с определенными сущностями, а с методами или стандартами. Например, управление версиями – типичное требование методологии для осуществления безопасности проекта.

Определение временной шкалы рабочего плана, включающей такие понятия, как пределы, периоды, сроки и т.д., есть важная стадия в разработке системы. Выбор этих шкал и связывание их с пакетами работ и задачами является спецификацией рабочего плана.

Мы различаем следующие классы задач. Задача разработки фактически включает опытно-конструкторские работы, которые также могут содержать другие действия, такие как моделирование или формальная проверка системы. Следуя предложенной на рис. 3 модели процесса системной инженерии, задача разработки сущности может начаться только после одобрения всех связанных с ней спецификаций.

Верификация определяется нами как процесс проверки не непосредственно реализации, а *задачи* разработки. Ее можно рассмотреть как аудит опытно-конструкторских работ с целью проверки *процесса* разработки. Верификация должна ответить на вопрос «разрабатывали ли мы систему правильно?». В программной инженерии типичными примерами верификации является соответствие правилам кодирования, управления версиями, нормам проектирования и т.д. Заметим, что верификация может иметь место только когда реализация системы была уже осуществлена, хотя она не исключает локальные проверки в процессе выполнения работ. Важным аспектом

является то, что верификация должна осуществляться группой инженеров, отличной от занимающихся непосредственно разработкой.

Тестовая задача проверяет (согласно тестовым случаям спецификаций) результаты разработки системы. Она осуществляется только после одобрения задачи верификации.

Наконец мы рассматриваем задачу валидации. Валидация проверяет результат реализации (после успешного завершения верификации и тестирования) на соответствие начальным требованиям. Валидация должна ответить на вопрос «разработали ли мы правильную систему?». Данная проверка направлена «сверху вниз» (т.е. от целостной системы к ее частям) и имеет целью интеграцию всех разработанных подсистем. Если валидация была успешной, продукт может быть «подписан» в реальное производство. Заметим, что на данном этапе некоторые свойства системы могут отличаться от тех, что были специфицированы изначально. Данный процесс мы называем характеризацией системы.

3 Разработка прототипа

Для проверки предложенного подхода и его применимости к различным ПрО нами был разработан инструмент OpenCookbook. Первая версия OpenCookbook была основана на Plone [14], а следующая на Drupal [15] CMS (Content Management System) системах управления содержанием с открытым исходным кодом. Переход на CMS Drupal был сделан нами по причине поддержки таксономии.

В обоих инструментах новый проект по разработке системы создается как web-портал с типами содержания, отражающими предложенную метаонтологию. Инструменты OpenCookbook позволяют осуществить связь между различными фазами процесса проектирования системы; выполнить проверку онтологии системы на непротиворечивость и полноту; сгенерировать документы, отражающими текущее описание; осуществить контроль версий и др.

Будучи web-инструментом, OpenCookbook позволяет организовать работу распределенных команд по определению будущей системы. OpenCookbook поддерживает следующие действия пользователей по определению системы, соответствующие предложенной модели SE:

– выдвижение требований;

– преобразование требований в спецификации (с определением нормальных, тестовых случаев, а также случаев отказа системы);

– архитектурную декомпозицию системы на сущности и их взаимодействия;

– определение пакетов работ и задач (разработки, валидации, верификации, тестирования).

Все эти действия поддерживаются общей БД, структура которой отражает предложенную метаонтологию SE. Для тестирования прототипа и проверки его применимости для различных ПрО нами было проведено множество экспериментов.

Наиболее важным испытанием прототипа было использование OpenCookbook для проектирования операционной системы реального времени OpenComRTOS [16] и сопутствующих инструментов. Формальные модели, используемые для разработки OpenComRTOS, имели очень высокий уровень абстракции, однако этот уровень полностью соответствовал предложенной метаонтологии SE.

Преимуществом использования OpenCookbook также был инкрементальный процесс проектирования OpenComRTOS. Начиная с небольшой и абстрактной модели, нами добавлялись все новые детали, пока не появилась модель, близкая к архитектуре реализации. Каждая промежуточная модель подвергалась проверке, что позволяло увидеть логические погрешности в проекте. Таким образом, модели разрабатывались малыми шагами и подвергались интенсивному анализу всеми членами команды (что

обусловлено фактом, что OpenCookbook есть web-инструмент). В результате уровень абстракции перешел в конкретную модель реализации. Такой подход также позволил нам обнаружить негативное воздействие знания принципов реализации, так как это заставляло инженеров думать в понятиях уже созданных ранее систем. Поэтому результат проектирования OpenComRTOS был более точным и имел более компактную системную архитектуру. Применение модели процесса параллельно со структурированием ПрО посредством онтологий позволило нам гарантировать правильность порядка шагов в процессе разработки системы.

Также наш подход мы сравнили с методом BPE (Business Process Engineering). Как результат мы обнаружили, что наша методология может быть более успешной благодаря использованию предметно-ориентированного подхода. Например, технический инженер может использовать CAD инструменты для моделирования различных сценариев использования системы; коммерческий директор может создать бизнес-план, моделируя бизнес-процесс с использованием финансовой терминологии и т.д. Это отражает тот факт, что в деловых кругах предназначение системы состоит в том, чтобы приносить доход, тогда как в инженерии предназначение системы состоит в обеспечении определенной функциональности.

В ходе всех этих экспериментов инструмент OpenCookbook был усовершенствован; однако в общем наши исследования доказали применимость предложенного подхода для моделирования систем в различных ПрО.

Выводы

В статье предлагается онтология для проектирования систем и основанная на ней модель процесса системной инженерии. Модель процесса разработки системы строится путем наложения ограничений на этапы определения системы. Таким образом, особенность подхода состоит в рассмотрении процесса системной инженерии как машины состояний. Предлагаемая онтология SE предназначена для проверки, разрабатываем ли мы *правильную* систему; модель процесса SE позволяет проверить, разрабатываем ли мы систему *правильно*. Для оценки предложенного подхода был разработан web-ориентированный инструмент OpenCookbook. Была осуществлена поверка применимости OpenCookbook для моделирования систем из различных предметных областей.

Наши дальнейшие исследования будут посвящены расширению класса методов, применимых к онтологии системной инженерии. Подвергнется формализации и дальнейшему изучению модель процесса SE, построенная на основе конечного автомата. Формализация метода моделирования процесса SE позволит разработчику создать собственный процесс проектирования систем, отражающий особенности конкретной организации. Подвергнутся дальнейшей разработке методы перехода между различными онтологиями SE, а также их связь с процессом определения системы.

Литература

1. Gruber T.R. A translation approach to portable ontologies / T.R. Gruber // Knowledge Acquisition. – 1993. – 5(2). – P. 199-220.
2. Alexander Kossiakoff Systems Engineering Principles and Practice / Alexander Kossiakoff and William N. - Wiley-Interscience. – 2002. - 488 p.
3. Benjamin S. Fabrycky. Systems Engineering and Analysis / Benjamin S. Blanchard and Wolter J. Fabrycky. – (5th Edition). – Prentice Hall, 2010. - 800 p.
4. Joseph E. Kasser. A Framework for Understanding Systems Engineering / Joseph E. Kasser. – Book-Surge Publishing. – 2007. – 378 p.

5. Mark Austin. PaladinRM: graph-based visualization of requirement organized for team-based design / Austin Mark, Mayank Vimal, and Shmunis Natalia // System engineering. – 2006. – Vol. 9, № 2. – P. 129.
6. Tim Weilkiens. Systems Engineering with SysML/UML. Modeling, Analysis, Design / Tim Weilkiens. – Morgan Kaufmann Publishers Inc. – [1st ed.]. – 2008. – 320 p.
7. [Электронный ресурс]. – Режим доступа : http://www.uml-forum.com/
8. [Электронный ресурс]. – Режим доступа : http://www.omgsysml.org
9. Berners-Lee T. The Semantic Web / Berners-Lee T., Hendler J., and Lassila O. // Scientific American. – May. – 2001.
10. [Электронный ресурс]. – Режим доступа : http://suo.ieee.org/
11. [Электронный ресурс]. – Режим доступа : http://www.jfsowa.com/ontology/toplevel.htm
12. [Электронный ресурс]. – Режим доступа : http://www.cyc.com/cyc-2-1/cover.html
13. OWL Web Ontology Language Guide. [Электронный ресурс]. – Режим доступа : http://www.w3.org/TR/owl-guide/
14. [Электронный ресурс]. – Режим доступа : www.plone.org
15. [Электронный ресурс]. – Режим доступа : www.drupal.org
16. OpenComRTOS: A Runtime Environment for Interacting Entities / Bernhard H.C. Sputh, Oliver Faust, Eric Verhulst, and Vitaliy Mezhuyev // Communicating Process Architectures 2009. – IOS Press. – 2009. – P. 173-184.

В.І. Межуєв

Онтологічні моделі систем та процесу системної інженерії

У статті будуються онтології систем, що охоплюють різні аспекти системної інженерії, включаючи формулювання вимог і специфікацій, архітектурне моделювання, створення робочого плану, тестування, валідацію й верифікацію систем. Особливістю підходу є застосування теорії автоматів для побудови онтологічної моделі процесу розробки систем. Розглядається практична реалізація запропонованого підходу в інструменті OpenCookbook.

V.I. Mezhuyev

Ontological Models of Systems and System Engineering Process

The ontologies of systems are developed in the paper. The proposed ontologies envelop various aspects of system engineering, including capturing requirements and specifications, architectural modelling, creation of a work plan, testing, validation and verification of systems. The feature of approach is development of the ontological model of system engineering process. Practical implementation of the proposed approach in the OpenCookbook tool is considered.

Статья поступила в редакцию 21.06.2010.